

# A Memetic Algorithm for the Minimum Conductance Graph Partitioning Problem

[working paper]

David Chalupa

Computer Science  
School of Engineering and Computer Science  
University of Hull  
Cottingham Road  
Hull HU6 7RX, UK  
Email: D.Chalupa@hull.ac.uk

**Abstract.** The minimum conductance problem is an NP-hard graph partitioning problem. Apart from the search for bottlenecks in complex networks, the problem is very closely related to the popular area of network community detection. In this paper, we tackle the minimum conductance problem as a pseudo-Boolean optimisation problem and propose a memetic algorithm to solve it. An efficient local search strategy is established. Our memetic algorithm starts by using this local search strategy with different random strings to sample a set of diverse initial solutions. This is followed by an evolutionary phase based on a steady-state framework and two intensification subroutines. We compare the algorithm to a wide range of multi-start local search approaches and classical genetic algorithms with different crossover operators. The experimental results are presented for a diverse set of real-world networks. These results indicate that the memetic algorithm outperforms the alternative stochastic approaches.

**Keywords.** conductance, graph partitioning, memetic algorithms, complex networks, pseudo-Boolean functions

## 1 Introduction

The minimum conductance problem finds its applications in *network community detection* [25], as well as in general *graph clustering* [6, 41]. It is also related

to other graph partitioning problems such as clique covering [8]. In the minimum conductance problem, the aim is to divide the vertex set into two subsets such that the “relative connectivity” of these two subsets is minimised. In some studies, it is also referred to as the *sparsest cut problem* [20].

Figure 1 illustrates the problem for a small 52-vertex social network. The problem aims at partitioning the vertex set into two partitions, minimising the ratio of the number of edges connecting vertices in different partitions to the number of all edges incident to the vertices incident with vertices of one of the partitions.

The problem was shown to be NP-hard by Šíma and Schaeffer approximately a decade ago [47]. Despite this fact, the problem has been overlooked as an optimisation problem. Leskovec et al used the conductance metric to evaluate the quality of network communities obtained by different types of algorithms [24, 25]. In this context, conductance is used as a measure of how well a community is separated from the rest of the graph.

The minimum conductance problem can naturally be represented as a 0-1 optimisation problem, in the form of a pseudo-Boolean function [12]. Conductance is defined for any partitioning, apart from those, which contain a partition consisting solely of isolated vertices. In the following, we will show that conductance can be expressed as a non-linear pseudo-Boolean function.

This outlines a link between this problem and a wide range of prominent problems in non-linear pseudo-Boolean optimisation such as maxSAT [52], maxkSAT [51], NK-landscapes [9, 54] or other tunable rugged objective functions [28].

For many of these problems, efficient partition crossovers [46], neighbourhood search strategies [11] and landscape analysis techniques have been applied [34, 50]. These results generally rely on the  $k$ -bounded nature of the pseudo-Boolean functions, i.e. these functions can be expressed as a sum of terms such that each term depends on at most  $k$  Boolean inputs [53]. In this paper, we lay the foundations for a similar study of the minimum conductance problem.

We have outlined that the minimum conductance problem is closely related to community detection. Over the years, many different strategies of community detection have been identified [4, 5, 15, 16, 18, 27, 33, 36]. Community structure of networks in many domains has been studied [59], including *social and biological networks* [17]. Related but more general concepts include graph clustering [40, 41, 43] and graph mining [1, 7]. These concepts find their applications in various areas, including *social media* [35], *web communities* [14], *cyberattack detection* [30], as well as *functional module detection* for protein-protein interaction networks [37].

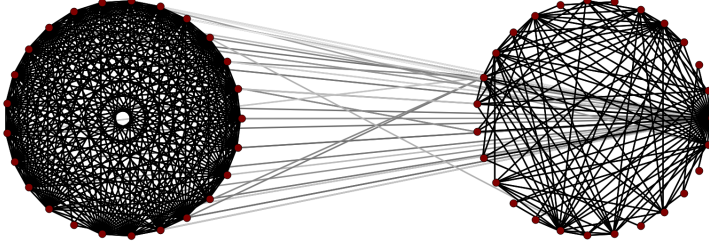


Figure 1: An example of a low conductance graph partitioning found for a 52-vertex social network sample *soc\_52*. The conductance value for this partitioning is 0.13108614. The drawing shows that the solution partitions the network into two dense clusters, connected relatively sparsely.

This problem is also closely related to the analysis of *scale-free* [2, 13] and *small-world networks* [48], as well as the *hierarchy* observed in real-world networks [31, 38].

*Contributions.* In this paper, we propose a *steady-state adaptive memetic algorithm* (StS AMA) for the minimum conductance problem. We also explore the potential of classical genetic algorithms and several local search strategies in solving the problem. We also derive an *efficient neighbourhood exploration strategy*, along with an efficient way of conductance recalculation in local search. Our experimental results indicate that StS AMA outperforms genetic algorithms with both one-point and uniform crossovers, as well as local search and randomised local search strategies. The experiments are provided using six different search algorithms applied to both *social networks* and *protein-protein interaction networks*.

In Section 2, we describe the minimum conductance problem and conductance as a pseudo-Boolean function. In Section 3, we describe our efficient neighbourhood exploration strategy and provide an overview of local search and crossover-based evolutionary algorithms we have used to tackle the problem. In Section 4, we describe our StS AMA. In Section 5, we present the obtained experimental results. Last but not least, in Section 6, we conclude the work and provide a discussion on its outcome and related open problems.

## 2 The Minimum Conductance Problem

In this section, we first describe the minimum conductance problem as a graph problem. Secondly, we formulate conductance as a pseudo-Boolean function

and discuss the ways for its efficient calculation and recalculation in local search and evolutionary algorithms.

## 2.1 Problem Definition

Let  $G = [V, E]$  be a connected undirected graph and let  $S \subset V$  be a subset of its vertices such that  $S \neq \emptyset$  and  $S \neq V$ . Then, the *volume* of this subset  $S$  is defined as follows:

$$Vol(S) = \sum_{v \in S} deg(v), \quad (1)$$

where  $deg(v)$  is the degree of  $v \in V$ , i.e. the number of its neighbours.

Then, the *conductance*  $\Phi(S)$  of a partitioning of  $V$  into the sets  $S$  and  $V \setminus S$  is defined using the following formula:

$$\Phi(S) = \frac{c_G(S)}{\min\{Vol(S), Vol(V \setminus S)\}}. \quad (2)$$

where

$$c_G(S) = \sum_{v \in S} deg_{V \setminus S}(v) = \sum_{v \in V \setminus S} deg_S(v), \quad (3)$$

with  $deg_{V \setminus S}(v) = |\{\{v, v'\} : v' \in V \setminus S\}|$  being the number of neighbours of  $v$  in the set  $V \setminus S$ .

One can see that  $\Phi(S)$  is not defined if  $Vol(S) = 0$  or  $Vol(V \setminus S) = 0$ . For a connected graph, that holds if and only if  $S = \emptyset$  or  $S = V$ .

For any other solutions, we have that  $Vol(S) > 0$  and  $Vol(V \setminus S) > 0$ . This implies that  $\Phi(S)$  can be rewritten as:

$$\Phi(S) = \max \left\{ \frac{c_G(S)}{Vol(S)}, \frac{c_G(S)}{Vol(V \setminus S)} \right\}. \quad (4)$$

## 2.2 Conductance as a Pseudo-Boolean Function and its Properties

Variable  $c_G(S)$  represents the sum of the numbers of neighbours of vertices  $v \in S$ , which are in  $V \setminus S$  (or vice versa). Hence,  $\Phi(S)$  can be further transformed into the following form:

$$\Phi(S) = \max \left\{ \frac{\sum_{v \in S} deg_{V \setminus S}(v)}{\sum_{v \in S} deg(v)}, \frac{\sum_{v \in V \setminus S} deg_S(v)}{\sum_{v \in V \setminus S} deg(v)} \right\}. \quad (5)$$

This shows that the conductance represents a maximum of two pseudo-Boolean functions. In each of these functions, the numerator is a sum of values, each of which depends on  $\deg(v)$  variables, depending on the degree of  $v$ . However, the denominators influence all of the elements of the sum. With each move of a vertex from  $S$  to  $V \setminus S$ , or vice versa, the volumes of the sets are changed.

In other words, with each bit flip in a 0-1 representation of the problem, the values of all elements of the sums can be potentially be changed.

Let  $S' = S \cup \{v\}$  or  $S' = S \setminus \{v\}$  and let  $c = c_G(S') - c_G(S)$ . Then, an improvement or a stagnation (i.e.  $\Phi(S') \leq \Phi(S)$ ) will be obtained if and only if:

$$\frac{\min\{Vol(S), Vol(V \setminus S)\}}{\min\{Vol(S'), Vol(V \setminus S')\}} \Phi(S) - \Phi(S) + \frac{c}{\min\{Vol(S'), Vol(V \setminus S')\}} \leq 0, \quad (6)$$

which can further be transformed to:

$$\min\{Vol(S), Vol(V \setminus S)\} - \min\{Vol(S'), Vol(V \setminus S')\} + \frac{c}{\Phi(S)} \leq 0. \quad (7)$$

One can see that this condition can be fulfilled in multiple ways and does not seem to be as straightforward as for the case of  $k$ -bounded pseudo-Boolean optimisation [53]. In fact, the numerator of conductance is  $k$ -bounded, where  $k$  is equivalent to the maximum degree of our graph. However, the value in the denominator can be potentially changed in all  $|V|$  components.

Remarkably, for  $k$ -bounded functions, improving moves can be identified in  $\mathcal{O}(1)$  time without actually scanning the neighbourhood [11]. It is worth noting that for the minimum conductance problem, it is possible to find moves, for which

$$\frac{c}{\min\{Vol(S'), Vol(V \setminus S')\}} < 0, \quad (8)$$

in  $\mathcal{O}(1)$  time. However, the volume ratio  $\frac{\min\{Vol(S), Vol(V \setminus S)\}}{\min\{Vol(S'), Vol(V \setminus S')\}}$  seems to complicate the situation for the minimum conductance problem.

Therefore, it remains open whether improving moves can be found in  $\mathcal{O}(1)$  time for minimum conductance problem. However, we will show in the next section that it is possible to recalculate the conductance for a single move in  $\mathcal{O}(1)$  time. This leads to efficient randomised local search strategies, as well as systematic local search, which can be used to scan the neighbourhood fully in  $\mathcal{O}(n)$  time.

### 3 Local Search Strategies and Genetic Algorithms

In this section, we first present our neighbourhood exploration strategy. Next, we describe the three local search and two genetic algorithms that we used in our experimental evaluations. The next section will then follow up with the description of StS AMA.

#### 3.1 Neighbourhood Exploration Strategy

The conductance for a partitioning  $S$  can be computed in  $\mathcal{O}(m)$  time for a graph on  $m$  edges, by iterating over the edges of the graph. At the same time, the current value of  $c_G(S)$ , and degrees  $\deg_S(v)$  and  $\deg_{V \setminus S}(v)$  can be calculated and stored as auxiliary data.

Let  $S' = S \cup \{v\}$ . Then, the following formulas can be used to recalculate the conductance in  $\mathcal{O}(1)$  time:

$$c_G(S') = c_G(S) - \deg_S(v) + \deg_{V \setminus S}(v), \quad (9)$$

$$\text{Vol}(S') = \text{Vol}(S) - \deg(v), \quad (10)$$

$$\text{Vol}(V \setminus S') = \text{Vol}(V \setminus S) + \deg(v). \quad (11)$$

If the move is accepted, then for each neighbour  $w$  of our vertex  $v$ , the auxiliary degrees can be updated as follows:

$$\deg_{S'}(w) = \deg_S(w) - 1, \quad (12)$$

$$\deg_{V \setminus S'}(w) = \deg_{V \setminus S}(w) + 1. \quad (13)$$

#### 3.2 Local Search and Genetic Algorithms

Many local search approaches may be used to solve this type of a problem [19], even though it seems that none of them have been applied yet. In the following, we describe the algorithms we have used to tackle the minimum conductance problem.

*Local search algorithm LS<sup>1</sup>.* This is a simple steepest descent search algorithm, which uses the strategy described above to test each of the possible bit flips in  $\mathcal{O}(1)$  time per vertex. The algorithm starts with a random bit string and chooses the best bit flip in each iteration. It stops whenever the best solution in the neighbourhood is not better than the current solution. Therefore, LS<sup>1</sup>

Algorithm 1: An Adaptive Local Search Algorithm (ALS<sup>1</sup>) for the Minimum Conductance Graph Partitioning Problem

	Output: best configuration $S_{best}$ found
1	$p_s = 1/2, \Phi_{best} = \infty$
2	while stopping criteria are not met
3	set each bit of $S$ to 1 with probability $p_s$
4	improve $S$ using LS <sup>1</sup> until the local optimum is reached
5	if $\Phi(S) \leq \Phi_{best}$
6	$\Phi_{best} = \Phi(S)$
7	$p_s = p_s/2$
8	else
9	$p_s = 1/2$
10	return $S_{best}$

guarantees that a local optimum is reached. If a local optimum is found, the local search is restarted.

*Adaptive local search algorithm ALS<sup>1</sup>.* This is an extension of LS<sup>1</sup>. The pseudocode of ALS<sup>1</sup> is given in Algorithm 1. The adaptive component relies on a gradual lowering of the probability of a 1-bit being generated in the initial solution. This way, we ensure that the algorithm also searches for asymmetric partitionings, which may be hard to reach if the initial solution contains 0-bits and 1-bits with equal probabilities. This is accomplished by starting with generating the initial solution by assigning 1-bits with probability  $p_s = 1/2$ . Then, in the next restart of the local search,  $p_s$  is halved to  $1/4$ . If the best solution sampled by the last run of the local search has not improved the best solution found so far,  $p_s$  is reset to  $1/2$ .

*Adaptive randomised local search algorithm ARLS<sup>1,2</sup>.* This algorithm is a randomised local search approach, which allows both moves of a single vertex between partitions, as well as exchanges of vertices between them. In each iteration, a move of one vertex or a move of two different vertices is tested using the approach described above. The move is accepted if the new conductance is at least as good as the current conductance. Both the test of a move and an update after the move take  $\mathcal{O}(\Delta)$  time, where  $\Delta$  is the maximum degree of a vertex.

*Adaptive genetic algorithm with one-point crossover AGA-1PX.* AGA-1PX is a relatively standard variant of a genetic algorithm with a one-point crossover. It uses a population  $P$  of  $p$  individuals initially generated at random. A 1-bit is placed in the initial solution with probability  $p_s$  and a 0-bit is placed into it with probability  $1 - p_s$ . The initial value  $p_s$  is set to  $1/2$ . Similarly to ALS<sup>1</sup>,  $p_s$  is iteratively halved until no improvement in the initial conductance

is obtained. This occurs whenever  $10^6$  candidate solutions have been generated without improvement of the best solution found so far. AGA-1PX then restarts the search with halved  $p_s$ . In the evolutionary phase, tournament of size  $t$  is used to select two parents. Two offspring are then generated using the one-point crossover. Next, mutation is performed on each offspring. Each bit in the solution is flipped with probability  $1/n$ , where  $n$  is the number of vertices in the graph. This is repeated until  $p$  new solutions are generated. We use a slightly elitist replacement strategy. All of the individuals are replaced with the offspring, apart from the best individual in the population.

*Adaptive genetic algorithm with uniform crossover AGA-UX.* This algorithm has almost exactly the same structure as AGA-1PX. The only difference is that for each pair of parents, one offspring is generated by the uniform crossover. In the uniform crossover, each bit is taken from the first parent with probability  $1/2$  and from the second parent otherwise [44]. The selection, mutation and replacement strategies are exactly the same as in AGA-1PX.

## 4 Steady-state Adaptive Memetic Algorithm

In this section, we introduce the steady-state adaptive memetic algorithm (StS AMA) that we propose for the problem.

Memetic algorithms have been widely used and their efficiency is related to properties of the fitness landscape [10, 22, 29, 42]. Memetic algorithms have also been previously used to solve different graph partitioning problems [3].

The pseudocode of our StS AMA is given in Algorithm 2. The algorithm uses a population of individuals, which represent local optima. In the beginning, the initial population is generated adaptively at random, and is improved by the application of the systematic local search algorithm  $LS^1$ . The adaptive component addresses the observation that low-conductance partitionings of real-world networks often seem to have an unbalanced structure. Therefore, each initial individual  $P_i$  is generated uniformly at random, with each bit set to 1 with probability  $p_s = 1/2$  and to 0 otherwise. As the next step,  $p_s$  is lowered to  $1/4$  to allow the exploration of unbalanced solutions. This process is described in steps 2-8 and stops whenever another lowering of  $p_s$  leads to a worse solution than the previous solution found.

This is followed by an evolutionary procedure. In each generation, two parents  $P_1$  and  $P_2$  are chosen by a tournament of size  $t$  in step 11. In step 12, their offspring  $O_1$  and  $O_2$  are created using one-point crossover. In steps 13-14,  $RLS^{1,2}$  and  $LS^1$  are consecutively applied to improve  $O_1$  and  $O_2$ . Note that after step 14, both  $O_1$  and  $O_2$  represent locally optimal partitionings. In steps 15-16,  $O_1$  and  $O_2$  are used to replace the worst individuals in the population. This process is iterated until a stopping criterion is met. In our experiments, we will simply use a time limit as this criterion.



Algorithm 2: A Steady-state Adaptive Memetic Algorithm (StS AMA) for the Minimum Conductance Graph Partitioning Problem

	Input: population size $p$ , tournament size $t$ , local search length $l$ Output: best configuration $P_{best}$ found
1	for $i = 1 \dots p$
2	$p_s = 1/2$
3	do
4	set each bit of a candidate for individual $P_i$ to 1 with probability $p_s$
5	improve the candidate for individual $P_i$ using LS <sup>1</sup> until the local optimum is reached
6	$p_s = p_s/2$
7	while the current candidate for $P_i$ is at least as good as the best of the previous candidates
8	set the best candidate sampled in steps 3-7 as the individual $P_i$
9	$P = \{P_1, P_2, \dots, P_p\}$
10	while stopping criteria are not met
11	pick parents $P_{p_1}$ and $P_{p_2}$ such that $p_1 \neq p_2$ using a tournament of size $t$
12	create the offspring $O_1$ and $O_2$ using one-point crossover
13	improve the offspring $O_1$ and $O_2$ using RLS <sup>1,2</sup> for $l$ iterations
14	improve the offspring $O_1$ and $O_2$ using LS <sup>1</sup> until the local optimum is reached
15	if $O_1 \notin P$ then replace the worst individual in $P$ with $O_1$
16	if $O_2 \notin P$ then replace the worst individual in $P$ with $O_2$
17	return the best individual $P_{best}$ in $P$

## 5 Experimental Results

In this section, we present the experimental of StS AMA and the other algorithms for several real-world network instances. We first describe the experimental protocol for our evaluation. Next, we present the result we obtained for social networks, protein-protein interaction networks and graph studied in network science literature.

### 5.1 Experimental Protocol

For the experimental evaluation, we have used real-world networks from three different sources. Firstly, we use samples of social networks, including public circles data from *Google+* and social network *Pokec* [45]. Next, we use protein-protein interaction networks from UCLA database of interacting proteins [39, 55, 56, 57]. Last but not least, we perform experiments also for several real-world networks studied in network science literature.

All experiments have been conducted in short-running and long-running forms, with 1 minute and 15 minute time limits.  $LS^1$  and  $ALS^1$  have been used in a multi-start form, restarting whenever a local optimum has been reached.  $ARLS^{1,2}$  has also been used in a multi-start form, with a restart being used after  $10^6$  iterations without improvement.

AGA-IPX and AGA-UX have been used with population size  $p = 100$  and tournament size  $t = 2$ . A restart has also been used if  $10^6$  individuals without improvement of the best solution found so far have been generated. This restart is also accompanied by adaptation of  $p_s$ .

StS AMA has also been used with population size  $p = 100$  and tournament size  $t = 2$ . Within StS AMA,  $RLS^{1,2}$  has been used with a maximum of  $l = 10^6$  iterations.

### 5.2 Results for Social Networks

In Table 1 and Table 2, the results obtained for social network samples are presented. The first column of these tables identifies the graph, followed by columns denoting the algorithm used, minimum and average conductance  $\Phi(S)$  obtained, as well as the success rate, i.e. the number of runs obtaining the best result out of all runs of the algorithm.

We have used the algorithms to solve the problem in a small 52-vertex social network sample *soc\_52*, samples of public circles data from social network *Google+*, including *gplus\_200*, *gplus\_500* and *gplus\_2000*, as well as samples of social network *Pokec*, including *pokec\_500*, *pokec\_2000*. A larger snapshot

Table 1: Comparison of the multi-start variants of the local search algorithms  $LS^1$ ,  $ALS^1$ ,  $ARLS^{1,2}$ , genetic algorithms AGA-1PX and AGA-UX, and StS AMA for the social networks in short runs with a 1 minute time limit.

G	algorithm	min $\Phi(S)$	$E[\Phi(S)]$	success rate
<i>soc_52</i>	$LS^1$	<b>0.13108614</b>	<b>0.13108614</b>	100 / 100
	$ALS^1$	<b>0.13108614</b>	<b>0.13108614</b>	100 / 100
	$ARLS^{1,2}$	<b>0.13108614</b>	<b>0.13108614</b>	100 / 100
	AGA-1PX	<b>0.13108614</b>	<b>0.13108614</b>	100 / 100
	AGA-UX	<b>0.13108614</b>	<b>0.13108614</b>	100 / 100
	StS AMA	<b>0.13108614</b>	<b>0.13108614</b>	100 / 100
<i>gplus_200</i>	$LS^1$	0.06158358	0.06158358	100 / 100
	$ALS^1$	<b>0.02040816</b>	0.06062967	2 / 100
	$ARLS^{1,2}$	0.04854369	0.08430937	1 / 100
	AGA-1PX	0.06158358	0.06411271	4 / 100
	AGA-UX	0.06158358	0.06453816	1 / 100
	StS AMA	<b>0.02040816</b>	<b>0.02551749</b>	84 / 100
<i>gplus_500</i>	$LS^1$	0.03688933	0.03688933	1 / 100
	$ALS^1$	0.03877551	0.04434687	1 / 100
	$ARLS^{1,2}$	0.04637097	0.07144919	1 / 100
	AGA-1PX	0.04608789	0.0716302	1 / 100
	AGA-UX	0.04772004	0.06875468	1 / 100
	StS AMA	<b>0.02040816</b>	<b>0.03293719</b>	11 / 100
<i>pokec_500</i>	$LS^1$	0.02744237	0.02952087	5 / 100
	$ALS^1$	<b>0.01345291</b>	0.02891339	4 / 100
	$ARLS^{1,2}$	0.03080082	0.05740674	1 / 100
	AGA-1PX	0.02757916	0.06173302	1 / 100
	AGA-UX	0.03593556	0.06307232	1 / 100
	StS AMA	<b>0.01345291</b>	<b>0.01345291</b>	100 / 100
<i>gplus_2000</i>	$LS^1$	0.06439536	0.07387353	1 / 100
	$ALS^1$	0.06540583	0.07352642	1 / 100
	$ARLS^{1,2}$	0.0625234	0.07842185	1 / 100
	AGA-1PX	0.08045977	0.1241733	1 / 100
	AGA-UX	0.07001123	0.11348164	1 / 100
	StS AMA	<b>0.0494713</b>	<b>0.05041461</b>	1 / 100
<i>pokec_2000</i>	$LS^1$	0.0346134	0.04484377	1 / 100
	$ALS^1$	0.02944942	0.04474195	1 / 100
	$ARLS^{1,2}$	0.0353544	0.06466346	1 / 100
	AGA-1PX	0.08342023	0.13448438	1 / 100
	AGA-UX	0.06587493	0.12499968	1 / 100
	StS AMA	<b>0.02360775</b>	<b>0.02521825</b>	3 / 100

Table 2: Comparison of the multi-start variants of the local search algorithms ALS<sup>1</sup>, ARLS<sup>1,2</sup>, and StS AMA for the social networks in longer runs with a 15 minute time limit.

$G$	algorithm	$\min \Phi(S)$	$E[\Phi(S)]$	success rate
<i>soc_52</i>	ALS <sup>1</sup>	<b>0.13108614</b>	<b>0.13108614</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.13108614</b>	<b>0.13108614</b>	30 / 30
	StS AMA	<b>0.13108614</b>	<b>0.13108614</b>	30 / 30
<i>gplus_200</i>	ALS <sup>1</sup>	<b>0.02040816</b>	0.06021106	1 / 30
	ARLS <sup>1,2</sup>	0.06233062	0.07979315	1 / 30
	StS AMA	<b>0.02040816</b>	<b>0.02178068</b>	29 / 30
<i>gplus_500</i>	ALS <sup>1</sup>	0.03777336	0.03972795	4 / 30
	ARLS <sup>1,2</sup>	0.04771372	0.06992862	1 / 30
	StS AMA	<b>0.034</b>	<b>0.03401609</b>	29 / 30
<i>pokec_500</i>	ALS <sup>1</sup>	0.02744695	0.02764317	13 / 30
	ARLS <sup>1,2</sup>	0.04347826	0.05789981	1 / 30
	StS AMA	<b>0.01345291</b>	<b>0.01345291</b>	30 / 30
<i>gplus_2000</i>	ALS <sup>1</sup>	0.06519208	0.0683073	1 / 30
	ARLS <sup>1,2</sup>	0.06232454	0.07828459	1 / 30
	StS AMA	<b>0.04941531</b>	<b>0.0499854</b>	6 / 30
<i>pokec_2000</i>	ALS <sup>1</sup>	0.03032428	0.03623088	1 / 30
	ARLS <sup>1,2</sup>	0.03238575	0.06166406	1 / 30
	StS AMA	<b>0.02360775</b>	<b>0.02463325</b>	7 / 30
<i>gplus_10000</i>	ALS <sup>1</sup>	0.07737166	0.08379242	1 / 30
	ARLS <sup>1,2</sup>	0.09005909	0.09886034	1 / 30
	StS AMA	<b>0.06645077</b>	<b>0.0671492</b>	1 / 30
<i>pokec_10000</i>	ALS <sup>1</sup>	0.05597898	0.08554391	2 / 30
	ARLS <sup>1,2</sup>	0.06784962	0.08064511	1 / 30
	StS AMA	<b>0.04513233</b>	<b>0.04656544</b>	1 / 30

of this network is also a part of the SNAP network dataset [23]. All of these social network samples are available in an anonymous form<sup>1</sup>.

For *soc\_52*, all algorithms have easily found the same low-conductance partitioning. However, the algorithms provided much more varied results for larger instances. Interestingly, ALS<sup>1</sup> and StS AMA performed significantly better for *gplus\_200* than the rest of the algorithms. This is most probably due to the adaptive mechanisms of these algorithms. This is also illustrated by Figure 2, showing the difference between the best partitioning found by most of the algorithms for *gplus\_200* and the partitioning found by ALS<sup>1</sup> and StS AMA. This highlights the role of the adaptive approach in the problem, since high-quality solutions to the problem often seem “asymmetric”. Additionally, the best partitioning found for *gplus\_500* has the same conductance value as the one found for *gplus\_200*. This indicates that the same community has been identified both in the smaller and the larger sample of the network. However, this solution has been found only by StS AMA for *gplus\_500*.

---

<sup>1</sup><http://davidchalupa.github.io/research/data/social.html>

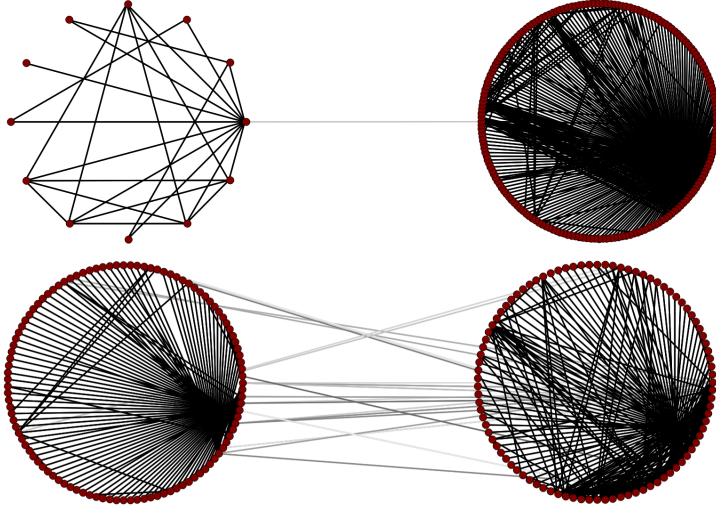


Figure 2: Illustrations of two low conductance graph partitionings found for the 200-vertex sample of publicly available circles data from Google+. The upper partitioning is highly unbalanced, with the conductance value 0.02040816. The lower partitioning represents a relatively balanced suboptimum with the conductance value 0.06158358. With the standard way of generating the initial solutions, the algorithms tend to converge to the suboptimum. It is the adaptive initial solution generation, which supports sampling of the unbalanced partitionings.

StS AMA has produced the best results also for *gplus\_2000*, *pokec\_500* and *pokec\_2000*. In addition, one can see that the best conductance has been obtained by StS AMA not only in the best runs, but also on average.

### 5.3 Results for Protein-Protein Interaction Networks

In Table 3 and Table 4, we present the results obtained for protein-protein interaction (PPI) networks. These experiments have been performed for the largest component of each of these networks, since unlike our social network samples, PPI networks do not have to be connected. Graphs with multiple connected components have a trivial solution with zero conductance.

PPI networks represent the data for the following species. *Celeg20160114* is a PPI network for *Caenorhabditis elegans*, *Dmela20160114* is for the fruit fly, *Ecoli20160114* is for *Escherichia coli*,

*Hpylo20160114* is for *Helicobacter pylori*, *Hsapi20160114* is a human PPI network, *Mmusc20160114* is for the house mouse, *Rnorv20160114* is for the brown rat and *Scere20160114* is a PPI network for a commonly used species of yeast.

Apart from *Rnorv20160114*, the results of different approaches for PPI networks seem to be much more varied. However, StS AMA has also been identified as by far the most efficient, providing the best results both in the best runs and on average. Intriguingly, the more generous time limit of 15 minutes seemed to give StS AMA improved and somewhat more stable performance. However, there still seems to be large space for performance improvement, either by using large parallel population, or by employing classical graph-theoretical tricks to discover promising regions of the search space.

## 5.4 Results for Network Science Graphs

Last but not least, Table 5 and Table 6 present the results obtained for graphs studied in network science literature. These graphs are taken from Newman’s network data repository<sup>2</sup>. Network *adjnoun* represents adjective-noun adjacencies in David Copperfield [32], *football* represents matches in a season of American college football league [17], *lesmis* is a network of character coappearances for Les Misérables [21], *zachary* is a network of friendships in a Karate club [58], *celegansneural* is the neural network for *Ceanorhabditis elegans* [49], *dolphins* is a social network of bottlenose dolphins [26] and *polbooks* is a network of political books.

For these instances, the algorithms obtained less varied results. No difference in performance of the algorithms was observed for *zachary* and *polbooks*. In addition, all algorithms obtained the best result also for the other instances. However, StS AMA performed better in terms of its success rate also for these instances.

## 6 Conclusions and Discussion

We proposed a *steady-state adaptive memetic algorithm* (StS AMA) for the *minimum conductance graph partitioning problem*. The algorithm combines the steady-state framework with two local search strategies. This includes both randomised local search and systematic local search to ensure that every solution in the population represents a local optimum. Both local search strategies are based on our own efficient *neighbourhood exploration strategy*.

The experimental results were presented for StS AMA, three local search algorithms (systematic local search algorithms LS<sup>1</sup>, ALS<sup>1</sup> and randomised local search algorithm ARLS<sup>1,2</sup>), as well as genetic algorithms with one-point

---

<sup>2</sup><http://www-personal.umich.edu/~mejn/netdata/>

Table 3: Comparison of the multi-start variants of the local search algorithms  $LS^1$ ,  $ALS^1$ ,  $ARLS^{1,2}$ , genetic algorithms AGA-1PX and AGA-UX, and StS AMA for the largest connected components of protein-protein interaction networks in short runs with a 1 minute time limit.

<i>G</i>	algorithm	$\min \Phi(S)$	$E[\Phi(S)]$	success rate
<i>Celeg</i> 20160114	$LS^1$	0.1360744	0.15048384	1 / 100
	$ALS^1$	0.08558559	0.13292784	1 / 100
	$ARLS^{1,2}$	0.13139222	0.14561728	1 / 100
	AGA-1PX	0.13117964	0.15636037	1 / 100
	AGA-UX	0.1341165	0.15054743	1 / 100
	StS AMA	<b>0.01226994</b>	<b>0.03331431</b>	29 / 100
<i>Dmela</i> 20160114	$LS^1$	0.30524601	0.32650744	1 / 100
	$ALS^1$	0.21733168	0.23160832	1 / 100
	$ARLS^{1,2}$	0.23824626	0.25221482	2 / 100
	AGA-1PX	0.38336905	0.38964162	1 / 100
	AGA-UX	0.32978997	0.33968028	1 / 100
	StS AMA*	<b>0.1559633</b>	<b>0.18935739</b>	1 / 100
<i>Ecoli</i> 20160114	$LS^1$	0.44194299	0.46075783	1 / 100
	$ALS^1$	0.35021218	0.38262331	1 / 100
	$ARLS^{1,2}$	0.31840414	0.32633622	1 / 100
	AGA-1PX	0.36671548	0.40512006	1 / 100
	AGA-UX	0.34065565	0.36573979	1 / 100
	StS AMA	<b>0.06060606</b>	<b>0.30657497</b>	1 / 100
<i>Hpylo</i> 20160114	$LS^1$	0.16543575	0.1878742	1 / 100
	$ALS^1$	0.16543575	0.18784113	1 / 100
	$ARLS^{1,2}$	0.17282127	0.1934479	1 / 100
	AGA-1PX	0.17429838	0.2037738	1 / 100
	AGA-UX	0.17429838	0.20141001	1 / 100
	StS AMA	<b>0.14899926</b>	<b>0.15361405</b>	1 / 100
<i>Hsapi</i> 20160114	$LS^1$	0.08230694	0.08867847	1 / 100
	$ALS^1$	0.06979472	0.08627573	1 / 100
	$ARLS^{1,2}$	0.07009483	0.08051043	1 / 100
	AGA-1PX	0.10188901	0.12269489	1 / 100
	AGA-UX	0.08241275	0.10327176	1 / 100
	StS AMA	<b>0.05024438</b>	<b>0.05558534</b>	1 / 100
<i>Mmusc</i> 20160114	$LS^1$	0.03706222	0.04354185	1 / 100
	$ALS^1$	0.03441296	0.04331399	1 / 100
	$ARLS^{1,2}$	0.03726083	0.04940749	1 / 100
	AGA-1PX	0.04620573	0.06044643	1 / 100
	AGA-UX	0.04432505	0.0587076	1 / 100
	StS AMA	<b>0.01428571</b>	<b>0.02361266</b>	1 / 100
<i>Rnorv</i> 20160114	$LS^1$	<b>0.00671141</b>	<b>0.00671141</b>	100 / 100
	$ALS^1$	<b>0.00671141</b>	<b>0.00671141</b>	100 / 100
	$ARLS^{1,2}$	<b>0.00671141</b>	0.0120047	34 / 100
	AGA-1PX	<b>0.00671141</b>	0.01036659	38 / 100
	AGA-UX	<b>0.00671141</b>	0.01179875	22 / 100
	StS AMA	<b>0.00671141</b>	<b>0.00671141</b>	100 / 100
<i>Scere</i> 20160114	$LS^1$	0.45503758	0.47404224	1 / 100
	$ALS^1$	0.23874941	0.24093083	1 / 100
	$ARLS^{1,2}$	0.23873166	0.24034586	1 / 100
	AGA-1PX	0.44679302	0.46746535	1 / 100
	AGA-UX	0.34502747	0.37700811	1 / 100
	StS AMA*	<b>0.23821699</b>	<b>0.23907947</b>	1 / 100

\* For these instances, a run of StS AMA took more than 1 minute due to the initial population sampling already taking more than 1 minute.

Table 4: Comparison of the multi-start variants of algorithms ALS<sup>1</sup>, ARLS<sup>1,2</sup> and StS AMA for the largest connected components of protein-protein interaction networks in long runs with a 15 minute time limit.

<i>G</i>	algorithm	$\min \Phi(S)$	$E[\Phi(S)]$	success rate
<i>Celeg</i> 20160114	ALS <sup>1</sup>	0.12004018	0.13222836	1 / 30
	ARLS <sup>1,2</sup>	0.12481645	0.14464208	1 / 30
	StS AMA	<b>0.01226994</b>	<b>0.01323653</b>	23 / 30
<i>Dmela</i> 20160114	ALS <sup>1</sup>	0.21853169	0.23147207	1 / 30
	ARLS <sup>1,2</sup>	0.2379034	0.24385345	1 / 30
	StS AMA	<b>0.12230216</b>	<b>0.17148753</b>	1 / 30
<i>Ecoli</i> 20160114	ALS <sup>1</sup>	0.3601381	0.38164997	1 / 30
	ARLS <sup>1,2</sup>	0.31797257	0.32312285	1 / 30
	StS AMA	<b>0.05714286</b>	<b>0.30108943</b>	1 / 30
<i>Hpylo</i> 20160114	ALS <sup>1</sup>	0.16543575	0.17767017	2 / 30
	ARLS <sup>1,2</sup>	0.17664449	0.19402062	1 / 30
	StS AMA	<b>0.14855876</b>	<b>0.15223137</b>	2 / 30
<i>Hsapi</i> 20160114	ALS <sup>1</sup>	0.07556946	0.08211216	1 / 30
	ARLS <sup>1,2</sup>	0.06647116	0.07461552	2 / 30
	StS AMA	<b>0.04076645</b>	<b>0.04409562</b>	1 / 30
<i>Mmusc</i> 20160114	ALS <sup>1</sup>	0.03658946	0.03945999	1 / 30
	ARLS <sup>1,2</sup>	0.03726083	0.04719777	1 / 30
	StS AMA	<b>0.01242028</b>	<b>0.01873836</b>	1 / 30
<i>Rnorv</i> 20160114	ALS <sup>1</sup>	<b>0.00671141</b>	<b>0.00671141</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.00671141</b>	0.01732172	13 / 30
	StS AMA	<b>0.00671141</b>	<b>0.00671141</b>	30 / 30
<i>Scere</i> 20160114	ALS <sup>1</sup>	0.23933513	0.24094871	1 / 30
	ARLS <sup>1,2</sup>	0.23846297	0.23913658	1 / 30
	StS AMA	<b>0.2376532</b>	<b>0.23781621</b>	2 / 30



Table 5: Comparison of the multi-start variants of the local search algorithms  $LS^1$ ,  $ALS^1$ ,  $ARLS^{1,2}$ , genetic algorithms AGA-1PX and AGA-UX, and StS AMA for the graphs studied in network science in short runs with a 1 minute time limit.

G	algorithm	$\min \Phi(S)$	$E[\Phi(S)]$	success rate
<i>adjnoun</i> [32]	$LS^1$	<b>0.27830179</b>	0.27867547	78 / 100
	$ALS^1$	<b>0.27830179</b>	0.27864151	80 / 100
	$ARLS^{1,2}$	<b>0.27830179</b>	0.29172815	2 / 100
	AGA-1PX	<b>0.27830179</b>	0.28204756	14 / 100
	AGA-UX	<b>0.27830179</b>	0.2854969	7 / 100
	StS AMA	<b>0.27830179</b>	<b>0.27863774</b>	82 / 100
<i>football</i> [17]	$LS^1$	<b>0.10116086</b>	<b>0.10116086</b>	100 / 100
	$ALS^1$	<b>0.10116086</b>	<b>0.10116086</b>	100 / 100
	$ARLS^{1,2}$	<b>0.10116086</b>	0.10267479	95 / 100
	AGA-1PX	<b>0.10116086</b>	0.11251169	56 / 100
	AGA-UX	<b>0.10116086</b>	0.11854058	38 / 100
	StS AMA	<b>0.10116086</b>	<b>0.10116086</b>	100 / 100
<i>lesmis</i> [21]	$LS^1$	<b>0.12252964</b>	<b>0.12252964</b>	100 / 100
	$ALS^1$	<b>0.12252964</b>	<b>0.12252964</b>	100 / 100
	$ARLS^{1,2}$	<b>0.12252964</b>	0.12256781	98 / 100
	AGA-1PX	<b>0.12252964</b>	0.12303712	73 / 100
	AGA-UX	<b>0.12252964</b>	0.12267752	92 / 100
	StS AMA	<b>0.12252964</b>	<b>0.12252964</b>	100 / 100
<i>zachary</i> [58]	$LS^1$	<b>0.12820513</b>	<b>0.12820513</b>	100 / 100
	$ALS^1$	<b>0.12820513</b>	<b>0.12820513</b>	100 / 100
	$ARLS^{1,2}$	<b>0.12820513</b>	<b>0.12820513</b>	100 / 100
	AGA-1PX	<b>0.12820513</b>	<b>0.12820513</b>	100 / 100
	AGA-UX	<b>0.12820513</b>	<b>0.12820513</b>	100 / 100
	StS AMA	<b>0.12820513</b>	<b>0.12820513</b>	100 / 100
<i>celegansneural</i> [49]	$LS^1$	<b>0.17575758</b>	<b>0.17575758</b>	100 / 100
	$ALS^1$	<b>0.17575758</b>	<b>0.17575758</b>	100 / 100
	$ARLS^{1,2}$	<b>0.17575758</b>	0.17894605	37 / 100
	AGA-1PX	<b>0.17575758</b>	0.18119584	30 / 100
	AGA-UX	<b>0.17575758</b>	0.18172543	41 / 100
	StS AMA	<b>0.17575758</b>	<b>0.17575758</b>	100 / 100
<i>dolphins</i> [26]	$LS^1$	<b>0.06382979</b>	<b>0.06382979</b>	100 / 100
	$ALS^1$	<b>0.06382979</b>	<b>0.06382979</b>	100 / 100
	$ARLS^{1,2}$	<b>0.06382979</b>	0.0761892	87 / 100
	AGA-1PX	<b>0.06382979</b>	<b>0.06382979</b>	100 / 100
	AGA-UX	<b>0.06382979</b>	<b>0.06382979</b>	100 / 100
	StS AMA	<b>0.06382979</b>	<b>0.06382979</b>	100 / 100
<i>polbooks</i>	$LS^1$	<b>0.04347826</b>	<b>0.04347826</b>	100 / 100
	$ALS^1$	<b>0.04347826</b>	<b>0.04347826</b>	100 / 100
	$ARLS^{1,2}$	<b>0.04347826</b>	<b>0.04347826</b>	100 / 100
	AGA-1PX	<b>0.04347826</b>	<b>0.04347826</b>	100 / 100
	AGA-UX	<b>0.04347826</b>	<b>0.04347826</b>	100 / 100
	StS AMA	<b>0.04347826</b>	<b>0.04347826</b>	100 / 100

Table 6: Comparison of the multi-start variants of algorithms ALS<sup>1</sup>, ARLS<sup>1,2</sup> and StS AMA for the graphs studied in network science in long runs with a 15 minute time limit.

G	algorithm	min $\Phi(S)$	$E[\Phi(S)]$	success rate
<i>adjnoun</i> [32]	ALS <sup>1</sup>	<b>0.27830179</b>	<b>0.27830179</b>	30 / 30
	ARLS <sup>1,2</sup>	0.28	0.29481774	1 / 30
	StS AMA	<b>0.27830179</b>	0.2784717	27 / 30
<i>football</i> [17]	ALS <sup>1</sup>	<b>0.10116086</b>	<b>0.10116086</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.10116086</b>	0.10270889	28 / 30
	StS AMA	<b>0.10116086</b>	<b>0.10116086</b>	30 / 30
<i>lesmis</i> [21]	ALS <sup>1</sup>	<b>0.12252964</b>	<b>0.12252964</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.12252964</b>	<b>0.12252964</b>	30 / 30
	StS AMA	<b>0.12252964</b>	<b>0.12252964</b>	30 / 30
<i>zachary</i> [58]	ALS <sup>1</sup>	<b>0.12820513</b>	<b>0.12820513</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.12820513</b>	<b>0.12820513</b>	30 / 30
	StS AMA	<b>0.12820513</b>	<b>0.12820513</b>	30 / 30
<i>celegansneural</i> [49]	ALS <sup>1</sup>	<b>0.17575758</b>	<b>0.17575758</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.17575758</b>	0.18392339	7 / 30
	StS AMA	<b>0.17575758</b>	0.17575758	30 / 30
<i>dolphins</i> [26]	ALS <sup>1</sup>	<b>0.06382979</b>	<b>0.06382979</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.06382979</b>	0.07366368	25 / 30
	StS AMA	<b>0.06382979</b>	<b>0.06382979</b>	30 / 30
<i>polbooks</i>	ALS <sup>1</sup>	<b>0.04347826</b>	<b>0.04347826</b>	30 / 30
	ARLS <sup>1,2</sup>	<b>0.04347826</b>	<b>0.04347826</b>	30 / 30
	StS AMA	<b>0.04347826</b>	<b>0.04347826</b>	30 / 30

and uniform crossovers (AGA-1PX and AGA-UX). The experiments were performed on real-world networks, including social network samples, protein-protein interaction networks and graphs studied in network science literature.

These results identified StS AMA as the most robust strategy to solve the minimum conductance problem. We have also identified that the performance gap between StS AMA and the other algorithms seems to become wider as the instances get larger. The largest gaps have been identified for large social network samples with 2000 vertices, as well as for large protein-protein interaction networks.

However, several problems remain open. It is not yet clear whether improving moves can be identified in  $\mathcal{O}(1)$  time for the minimum conductance problem, similarly to max-SAT or NK-landscapes [51, 52]. Performance of StS AMA for large problem instances also suggests that improved results may be obtained by a parallel or a distributed variant of the algorithm. Last but not last, a hybrid approach, combining StS AMA with classical graph-theoretical techniques can also be a way to improve the performance of the algorithm.

## References

- [1] C. C. Aggarwal and H. Wang. *Managing and Mining Graph Data*. Springer, Berlin / Heidelberg, 2010.
- [2] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
- [3] U. Benlic and J. K. Hao. A multilevel memetic approach for improving graph k-partitions. *IEEE Transactions on Evolutionary Computation*, 15(5):624–642, 2011.
- [4] C. Biemann. Chinese whispers: An efficient graph clustering algorithm and its application to natural language processing problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 73–80, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.
- [5] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.
- [6] U. Brandes, M. Gaertler, and D. Wagner. Experiments on graph clustering algorithms. In *European Symposium on Algorithms*, pages 568–579. Springer, 2003.
- [7] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):Article No. 2, 2006.
- [8] D. Chalupa. Partitioning networks into cliques: A randomized heuristic approach. *Information Sciences and Technologies Bulletin of ACM Slovakia*, 6(3):1–8, 2014.
- [9] W. Chen, D. Whitley, D. Hains, and A. Howe. Second order partial derivatives for NK-landscapes. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, pages 503–510, New York, NY, USA, 2013. ACM.
- [10] X. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation*, 15(5):591–607, 2011.
- [11] F. Chicano, D. Whitley, and A. M. Sutton. Efficient identification of improving moves in a ball for pseudo-boolean problems. In Dirk V. Arnold, editor, *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*, pages 437–444, 2014.

- [12] Y. Crama, P. Hansen, and B. Jaumard. The basic algorithm for pseudo-boolean programming revisited. *Discrete Applied Mathematics*, 29(2):171–185, 1990.
- [13] S. N. Dorogovtsev and J. F. F. Mendes. Language as an evolving word web. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 268(1485):2603–2606, 2001.
- [14] G. W. Flake, S. Lawrence, and C. L. Giles. Efficient identification of web communities. In R. Ramakrishnan, S. Stolfo, R. Bayardo, and I. Parsa, editors, *Proceedings of the 6th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '00*, pages 150–160, New York, NY, 2000. ACM.
- [15] F. Folino and C. Pizzuti. An evolutionary multiobjective approach for community discovery in dynamic networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1838–1852, 2014.
- [16] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.
- [17] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
- [18] M. Gong, Q. Cai, X. Chen, and L. Ma. Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. *IEEE Transactions on Evolutionary Computation*, 18(1):82–97, 2014.
- [19] H. H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2004.
- [20] R. Khandekar, K. Hildrum, S. Parekh, D. Rajan, J. Wolf, K. L. Wu, H. Andrade, and B. Gedik. *COLA: Optimizing Stream Processing Applications via Graph Partitioning*, pages 308–327. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.
- [21] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA, 1993.
- [22] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488, 2005.
- [23] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. <http://snap.stanford.edu/data>, June 2014.

- 
- [24] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.
  - [25] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In J. Freire M. Rappa, P. Jones and S. Chakrabarti, editors, *Proceedings of the 19th international conference on World wide web, WWW '10*, pages 631–640, New York, NY, 2010. ACM.
  - [26] D. Lusseau, K. Schneider, O. J. Boisse, P. Haase, E. Slooten, and S. M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.
  - [27] L. Ma, M. Gong, J. Liu, Q. Cai, and L. Jiao. Multi-level learning based memetic algorithm for community detection. *Applied Soft Computing*, 19:121–133, 2014.
  - [28] N. Manukyan, M. J. Eppstein, and J. S. Buzas. Tunably rugged landscapes with known maximum and minimum. *IEEE Transactions on Evolutionary Computation*, 20(2):263–274, 2016.
  - [29] P. Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303–325, 2004.
  - [30] A. Metwally, D. Agrawal, and A. El Abbadi. Detectives: detecting coalition hit inflation attacks in advertising networks streams. In *Proceedings of the 16th international conference on World Wide Web*, pages 241–250. ACM, 2007.
  - [31] P. Náther, M. Markošová, and B. Rudolf. Hierarchy in the growing scale-free network with local rules. *Physica A*, 388:5036, 2009.
  - [32] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(036104):036104–1–036104–19, 2006.
  - [33] A. Noack. Energy models for graph clustering. *Journal of Graph Algorithms and Applications*, 11(2):453–480, 2007.
  - [34] G. Ochoa, F. Chicano, R. Tinós, and D. Whitley. Tunnelling crossover networks. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, pages 449–456. ACM, 2015.

- [35] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos. Community detection in social media. *Data Mining and Knowledge Discovery*, 24(3):515–554, 2012.
- [36] C. Pizzuti. A multiobjective genetic algorithm to find communities in complex networks. *IEEE Transactions on Evolutionary Computation*, 16(3):418–430, 2012.
- [37] C. Pizzuti and S. E. Rombo. Algorithms and tools for protein–protein interaction networks clustering, with a special focus on population-based stochastic methods. *Bioinformatics*, 30(10):1343–1352, 2014.
- [38] E. Ravasz and A. L. Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67:026112, 2003.
- [39] L. Salwinski, C. S. Miller, A. J. Smith, F. K. Pettit, J. U. Bowie, and D. Eisenberg. The database of interacting proteins: 2004 update. *Nucleic acids research*, 32(suppl 1):D449–D451, 2004.
- [40] S. E. Schaeffer. *Advances in Knowledge Discovery and Data Mining: 9th Pacific-Asia Conference, PAKDD 2005, Hanoi, Vietnam, May 18-20, 2005. Proceedings*, chapter Stochastic Local Clustering for Massive Graphs, pages 354–360. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
- [41] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
- [42] K. Sörensen and M. Sevaux. MA|PM: memetic algorithms with population management. *Computers & Operations Research*, 33(5):1214–1225, 2006.
- [43] D. A. Spielman and S. H. Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.
- [44] G. Syswerda. Uniform Crossover in Genetic Algorithms. In David J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9, San Mateo, California, 1989. Morgan Kaufmann Publishers, Inc.
- [45] L. Takac and M. Zabovsky. Data analysis in public social networks. In *International Scientific Conference and International Workshop Present Day Trends of Innovations*, pages 1–6, 2012.
- [46] R. Tintos, D. Whitley, and F. Chicano. Partition crossover for pseudo-boolean optimization. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, pages 137–149. ACM, 2015.

- 
- [47] J. Šíma and S. E. Schaeffer. On the NP-completeness of some graph cluster measures. In J. Wiedermann, G. Tel, J. Pokorný, M. Bieliková, and J. Štuller, editors, *Proceedings of the 32nd International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM '06*, pages 530–537, Berlin / Heidelberg, 2006. Springer.
- [48] D. J. Watts. *Small Worlds*. Princeton University Press, Princeton, NJ, 1999.
- [49] D. J. Watts and S. H. Strogatz. Collective dynamics of “small-world” networks. *Nature*, 393(6684):440–442, 1998.
- [50] D. Whitley. Mk landscapes, NK landscapes, MAX-kSAT: A proof that the only challenging problems are deceptive. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO '15*, pages 927–934, New York, NY, USA, 2015. ACM.
- [51] D. Whitley and W. Chen. Constant time steepest descent local search with lookahead for NK-landscapes and MAX-kSAT. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 1357–1364. ACM, 2012.
- [52] D. Whitley, A. Howe, and D. Hains. Greedy or not? Best improving versus first improving stochastic local search for MAXSAT. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, AAAI'13*, pages 940–946. AAAI Press, 2013.
- [53] L. D. Whitley, F. Chicano, and B. W. Goldman. Gray box optimization for Mk landscapes (NK Landscapes and MAX-kSAT). *Evolutionary computation*, 24(3):491–519, 2016.
- [54] A. H. Wright, R. K. Thompson, and J. Zhang. The computational complexity of NK fitness functions. *IEEE Transactions on Evolutionary Computation*, 4(4):373–379, 2000.
- [55] I. Xenarios, E. Fernandez, L. Salwinski, X. J. Duan, M. J. Thompson, E. M. Marcotte, and D. Eisenberg. Dip: the database of interacting proteins: 2001 update. *Nucleic acids research*, 29(1):239–241, 2001.
- [56] I. Xenarios, D. W. Rice, L. Salwinski, M. K. Baron, E. M. Marcotte, and D. Eisenberg. Dip: the database of interacting proteins. *Nucleic acids research*, 28(1):289–291, 2000.
- [57] I. Xenarios, L. Salwinski, X. J. Duan, P. Higney, S. M. Kim, and D. Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellular networks of protein interactions. *Nucleic acids research*, 30(1):303–305, 2002.

- [58] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.
- [59] X. S. Zhang, Z. Li, R. S. Wang, and Y. Wang. A combinatorial model and algorithm for globally searching community structure in complex networks. *Journal of Combinatorial Optimization*, 23(4):425–442, 2012.